Tampereen ammattikorkeakoulu

Pienpelikonsoli

5G00ET65-3008 Sulautetun järjestelmän ohjelmointi ja mikrokontrollerit

Hikarit

Aleksi Polso, Leona Suzuki & Nina Laaksonen

Projektityö
Joulukuu 2023

Tietotekniikan insinööri
22TietoB

SISÄLLYS

ERITYISSANASTO tai LYHENTEET JA TERMIT (valitse jompikumpi)

TAMK	Tampereen ammattikorkeakoulu
op	opintopiste
SJOM	Sulautetun järjestelmän ohjelmointi ja mikrokontrollerit
rauta	Tarkoitetaan tekniikan fyysistä osaa esim. virtapiiriä
Looppi tai silmukka	Ohjelman rakenne, jossa ohjelma toistaa valittuja asioita, kunnes määrätty tilanne saadaan aikaiseksi.
Nano tai Arduino	Arduino Nano on mikro-ohjainalusta, joka käyttää ATMega328-mikrokontrolleria. Se on suunniteltu helpottamaan elektroniikan ja ohjelmoinnin opiskelua sekä prototyyppien rakentamista.
TinkerCad	Sivusto, jolla voi testata kytkentöjen ja koodien toimintaa.
keskeytys	Ohjelman ajoon lisätty ajon keskeyttävä elementti, jonka aikana suoritetaan yksinkertaisia muutoksia ohjelmassa.
ylivuotokeskeytys	Ohjelmaan luotu keskeytys, joka tapahtuu ajastimeen asetetun maksimirajan tullessa täyteen. Tällöin tapahtuu ylivuoto ja ajastin alkaa alusta.
mikrokontrolleri	Siru, johon on valmiiksi luotu toiminnallisuuksia kytkennöillä ja ohjelmistolla. Nanossa ATMega328.
lohko	Mikrokontrollerissa oleva osa, jolla hallinnoi sen osan toiminnallisuutta. Esim. ajastin.
rekisteri, rekisteriohjaus	Lohkojen ohjauksen hallinta.
komponentti	Osa, jolla on toiminnallisuutta raudassa.
kytkin	Nappi. Tämä on myös komponentti.
adapteri	Sovitin, joka mahdollistaa erilaisten laitteiden, komponenttien, liittimien tai tekniikoiden yhdistämisen

1 MÄÄRITTELY JA TAVOITTEET

Tarkoituksena on tehdä pienpeliprojekti, jossa hyödynnetään Arduino Nanoa ja sen mikrokontrollerin ATMega328 erilaisia lohkoja ja ominaisuuksia. Tavoitteenamme on tehdä valikko, kolme eri peliä ja koteloida projekti kompaktiin laatikkoon, jotta kokonaisuus olisi siisti.

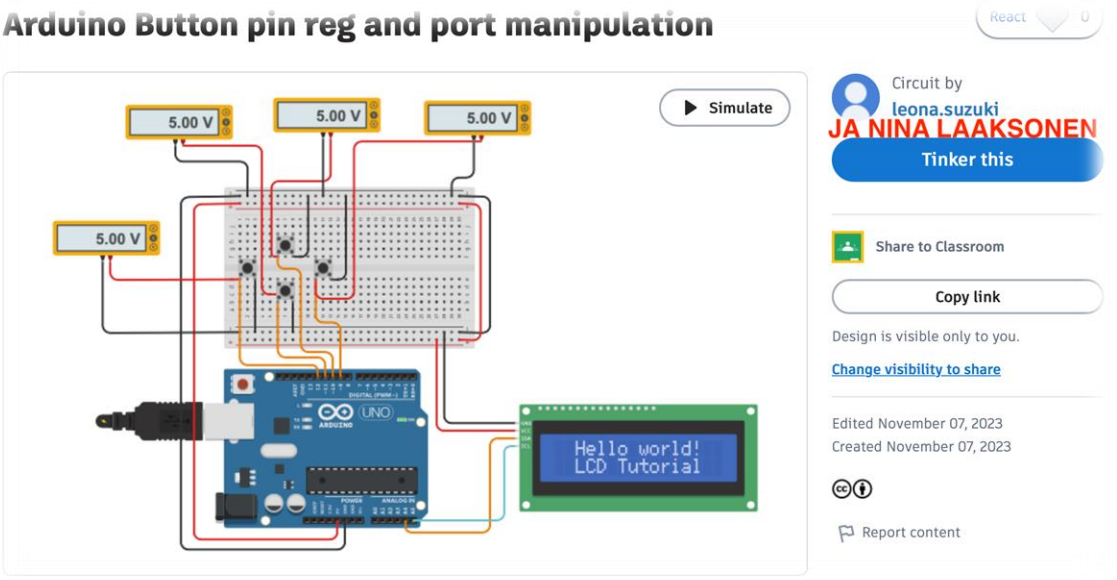
Arvosanatavoite projektista on 5. Tähän tarvitaan vähintään 4 mikrokontrollerin lohkoa, joista 3 käytetään rekistereiden kautta, jokin erityinen piirre, toimiva 2 suuntainen käyttöliittymä ja 2 eri keskeytystä.

Alkuperäisessä suunnitelmassa valitsimme EEPROM, T/C1, AD conv ja TWI lohkot käyttöön, joista käytimme rekistereillä T/C1 ja AD conv. Tarvittaviksi keskeytyksiksi valitsimme T/C1, eli TimerOne ja kytkinkeskeytyksen.

2 SUUNNITELMA

2.1 Miten toteutetaan

Määritetään ensin pelin ominaisuudet, jonka jälkeen kartoitetaan projektiin tarvittavat komponentit. Tämän jälkeen tarkistetaan, onko määritys tavoitteen mukainen. Suunnitellaan TinkerCad:lla ohjelmaa niin, paljon kun voidaan ennen kuin aloitetaan rakentaminen. Kun määritys, suunnittelu ja tavoitteet on tarkistettu, rakennamme alustan ja kytkennät, sen jälkeen ohjelmoimme toteutuksen.



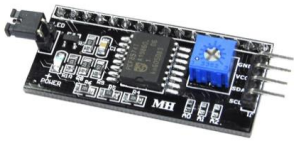
2.2 Työnjako

Leo tekee valikon, rekisteriohjauksen kytkinten käyttöön, Minuuttipelin ja kytkinten liittämisen komponentteina Arduinoon. Nina tekee Spede-pelin, keskeytykset, T/C1 rekisteriohjauksen ja ADC rekisteriohjauksen. Aleksi tekee villapaidan, Sakari-pelin ja kytkee valoanturin.

3 TOTEUTUS

3.1 Käytetyt komponentit

Komponenttilista

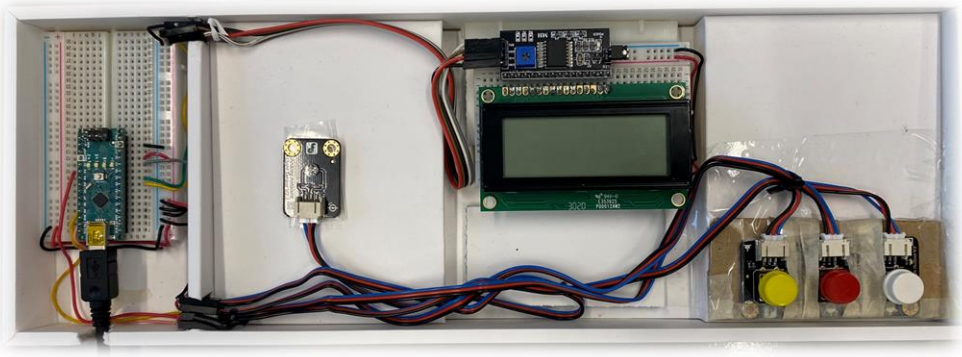
ARDUINO NANO, ARDUINO S.r.l	
DFRobot Ambient Light Sensor SKU DFR0026	
Gravity: Digital Push Button (Red)-DFRobot (DFR0029-R)	
Gravity: Digital Push Button (Yellow) (DFR0029-Y)	
Gravity:Digital Push Button (White) (DFR0029-W)	
MIDAS DISPLAYS MC42005A6W-BNMLWI-V2	
ARDUINO IIC/I2C INTERFACE LCD1602 ADAPTER	

Lisäksi:

- Noin 1m 20cm erivärisiä kytkentäjohtoa
- 4+1 komponenttien kytkentäjohtoa liittimillä
- USB-Mini
- Villapaita
- Kotelo valkoinen

3.2 Rauta sisältä

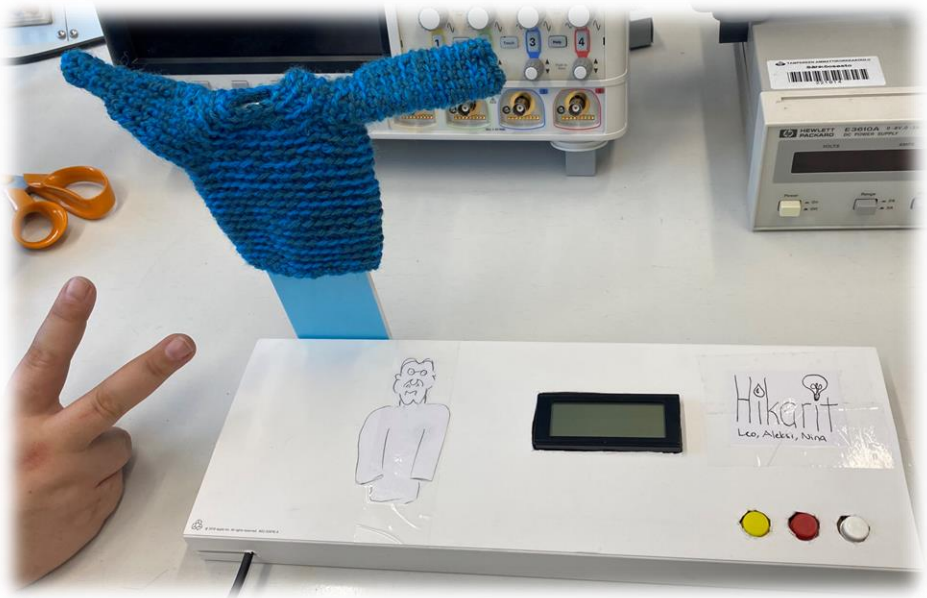
Neljärivinen LCD näyttö on kytketty I2C adapteria käyttäen kiinni Arduinoon, joka käyttää Arduinon mikrokontrollerissa TWI-lohkoa. Kytkimet ovat kytketty Arduinon digitaalisiin pinneihin: keltainen D9, valkoinen D11 ja punainen D3. Punainen kytkin on kytketty D3, jotta kytkinkeskeytyksen voi ottaa käyttöön. Tiimimme tavoite oli, että kytkentä toteutetaan mahdollisimman siististi. Myös lopulliseen työhön suunniteltiin, että koko rauta piilotetaan loppukäyttäjältä ja jätetään ainoastaan näkyville loppukäyttäjälle tarpeellinen osa.



Kuva 1. Rauta toteutettuna.

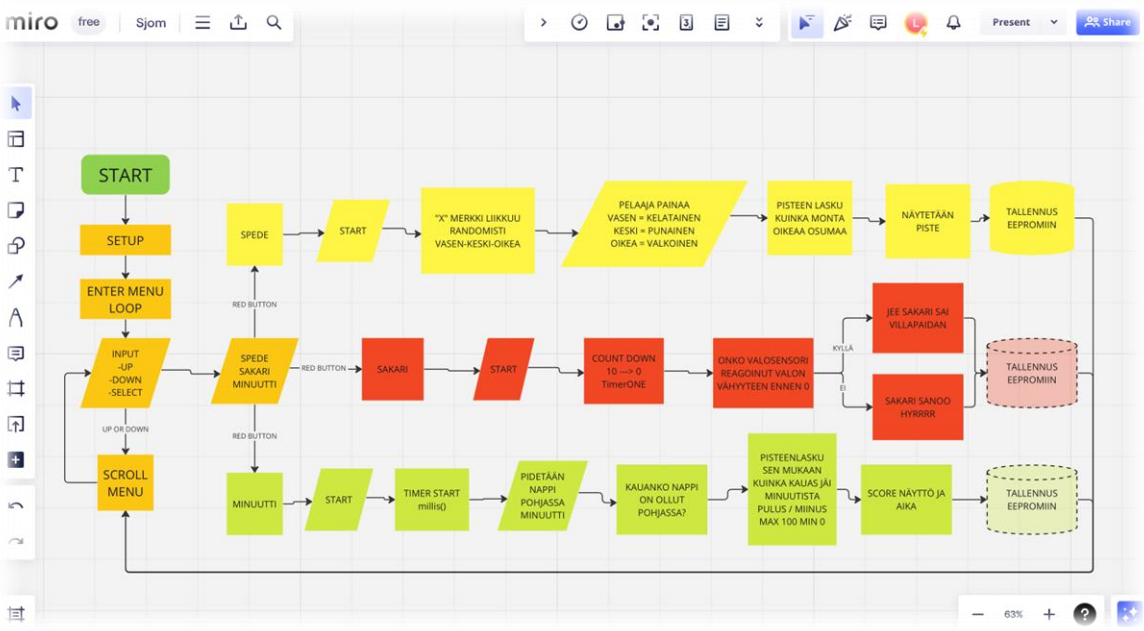
3.3 Rauta ulkoa

Lopullinen työ koteloitiin kokonaan valkoisen näppäimistön laatikon sisälle, jotta toteutus näyttäisi mahdollisimman siistiltä.



Kuva 2. Lopullinen tuote.

3.4 Ohjelma



Kuva 3. Ohjelman rakenne. Suunniteltu Mirolla.

3.4.1 Valikko

Valikon, joka on osa kaksisuuntaista käyttöliittymää, toteutus tehtiin niin, että se pyörii Nanon pääsilmukassa. Valikossa navigoidaan alaspäin valkoisella ja ylöspäin keltaisella kytkimellä. Kun halutun pelin kohdalla on tähdet ”*”, voidaan valita peli punaisella kytkimellä.

Kuva 4. Valikko.

3.4.2 Spede-peli

Spede-pelin alussa on lähtölaskenta kolmesta sekunnista alaspäin, jonka jälkeen itse peli alkaa. Pelissä X-merkki liikkuu sattumanvaraisesti oikealle, keskelle tai vasemmalle. Pelaajan tulee painaa kohtaa, jossa X on saadakseen pisteitä: vasemmalla on keltainen, keskellä on punainen ja oikealla on valkoinen.

Kuva 5. Spede-pelin lähtölaskenta.

Kuva 6. Spede-pelin peli

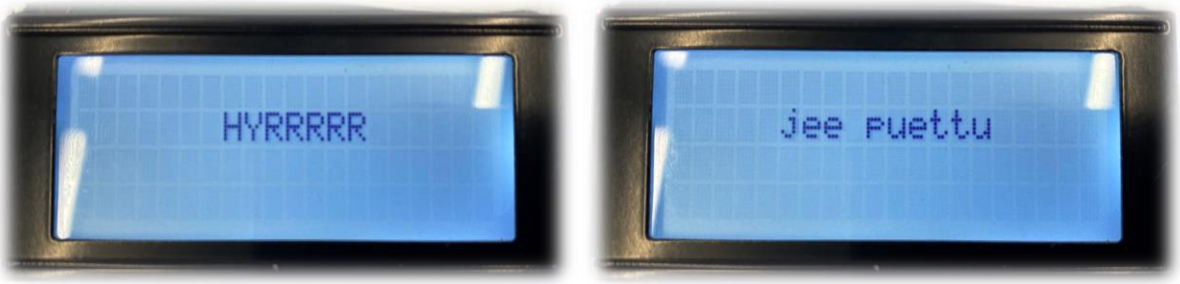
Spede-peli loppuu, jos käyttäjä painaa väärää kytkintä tai ohjelmaan lisätty Timer1-ylivuotokeskeytys aktivoituu. Pelin lopuksi näytetään käyttäjän saamat pisteet, voittiko käyttäjä ja parhaat kolme tulosta EEPROM-muistista. Jos saadut pisteet ovat kolmen parhaan joukossa, ne tallennetaan EEPROM-muistiin asianmukaiselle paikalle.

Kuva 7. Spede-pelin loppu: pisteitä 0 ja ne eivät ole kolmen parhaan joukossa.

3.4.3 Sakari-peli

Pelin käynnistyessä laskuri alkaa laskemaan kymmenestä nollaan. Jos pelaaja onnistuu pukemaan Sakarille villapaidan, eli peittää valosensorin, ennen kuin laskuri on nollassa, peli antaa palautteen ”jee puettu” ja mikäli pelaaja ei onnistu tehtävässä peli antaa palautteen ”HYRRRRR”.

Kuva 8. Sakari-peli käynnissä.



Kuva 9. Sakari-pelin häviö ja voitto.

Kuva 10. Sakari itse. Rinnan kohdalle on piilotettu valosensori.

3.4.4 Minuuttipeli

Valkoisella kytkimellä ohjattava peli, jossa on tarkoitus mitata käyttäjän sisäisellä kellolla minuutti. Peli alkaa painamalla valkoinen nappi pohjaan ja loppuu, kun nappi nostetaan ylös. Napin painamisen aika pisteytetään ja tulostetaan näytölle.

Kuva 11. Minuuttipelin alkutuloste.

Kuva 12. Minuuttipelissä valkoinen nappi on pohjassa ja aikaa lasketaan.

Kuva 13. Minuuttipelin loppu.

4 SELVITYS

4.1 Raudan testaukset

Jokainen kytkin on testattu yleismittarilla ja todettu sen toimivuus. Kytkintä painaessa vastus nousee niin suureksi, että jännite menee 0V, ja kytkimen ollessa vapaana se päästää virran läpi. Valoanturi on testattu toimivaksi vertaamalla lukuja valoisamman ja pimeämmän paikan välillä ja todettu ne oikeanlaisiksi. Näyttö on todettu toimivaksi silmämääräisesti sen toimiessa ja kokeiltu Wire-kirjaston avulla, onko sen IP-osoite oikea.

4.2 Valikon testaukset

Valikko etenee oletetusti kytkintä painettaessa. Kytkin liikuttaa useamman pykälän kerrallaan, kuten sen kuuluukin sitä pitkään painettaessa. Punaiseen kytkimeen lisätty keskeytys käynnistää jokaisen pelin oikealla vuorollaan.

4.3 Spede-pelin testaukset

Peliin käytettävät kytkimet valitsevat oikealla logiikalla reaktion ja mikään kytkin ei anna toistamiseen valintaa, jos nappia pitää pidemmän aikaa pohjassa. Pisteenlasku toteutuu oikein logiikan mukaisesti. T/C1 keskeytys toimii oletetusti: jokaisen pisteen, eli onnistuneen oikean kytkimen painamisen jälkeen, kytkimen painamiseen käytettävä aika pienenee. Pelin loputtua peli tulostaa pisteet oikein, riippuen siitä, onko pisteet EEPROM:iin tallennetuissa parhaassa kolmessa pisteessä. Parhaiden pisteiden saatua tallennuslogiikka toimii oikein.

4.4 Sakari-pelin testaukset

Pelin tulosteet näytölle toimivat oikein. Valoanturi reagoi sen peittämiseen voitolla ja peli loppuu, jos valoanturia ei peitetä.

4.5 Minuuttipelin testaukset

Peli käynnistyy suunnitellusti, valkoista kytkintä painettaessa pohjaan ajan lasku toimii oikein ja laskutoimitukset on varmistettu pisteenlaskussa.

5 POHDINTA

Alkuperäinen suunnitelma, jossa oli tähtien keräyspeli kahden valoanturin ohjaamana, vaihdettiin yksinkertaisempaan ja hauskempaan toteutukseen, Sakarin villapaitapeliin. Työhön saatiin silti tarvittavat ominaisuudet ja siitä saatiin hyvin toimiva.

Valitsimme alun perin D9, D10 ja D11 pinnit, joista jouduimme vaihtamaan myöhäisessä vaiheessa projektin tekoa D10 pinnin D3, jotta saimme kytkinkeskeytyksen toimimaan. Tämän olisi voinut välttää tarkemmalla suunnittelutyöllä alussa, josta opimme, että tarkka suunnittelutyö on tärkeää. Myös yleisten kytkinten ohjelmalla hallinnan käytäntöihin tarvittiin lisäapua opettajalta. Kun nämä saatiin selville, valmistui projekti nopeasti.

Pelien logiikassa olisi voinut myös hyväksikäyttää Arduinolle ominaista ”void loop()” aliohjelmaa enemmän. Nyt jokaisessa aliohjelmassa on oma while-silmukka, jolla sitä hallitaan, vaikka sen olisi voinut toteuttaa jo ajossa käytettävän silmukan kanssa. Tämä ei onneksi vaikuta käytettävyyteen tai toimivuuteen.

Sakari-pelissä olisi voitu toteuttaa niin, että se tunnistaisi vain peliin suunnitellun villapaidan. Nyt pelin voi suorittaa ihan millä vaan valoa läpäisemättömällä esineellä tai asialla. Mahdollinen parempi tapa olisi kytkeä villapaita tiettyyn vain siihen sopivaan liittimeen.

Pieniä ongelmia oli myös näytön Arduinoon kytkemisessä, koska johdot olivat jostain syystä väärinpäin. Hienosti toimii opettajan avustuksen jälkeen.

LÄHTEET

Moodle - SJOM kurssin kurssimateriaali. Arduinon pinnijärjestys. PDF.

Moodle - SJOM kurssin kurssimateriaali. ATMega datalehti. PDF.

Moodle - SJOM kurssin kurssimateriaali. AD muunnoksen esimerkkikoodi.

Moodle - SJOM kurssin kurssimateriaali. Timer1 esimerkkikoodi.

DFRobot opas projektissa käytettyyn analogiseen valosensoriin. Verkkosivu. 01.12.2023. https://wiki.dfrobot.com/DFRobot_Ambient_Light_Sensor_SKU_DFR0026

DFRobot opas projetissa käytettyihin kytkimiin. Verkkosivu. 01.12.2023. https://wiki.dfrobot.com/DFRobot_Digital_Push_Button_SKU_DFR0029

Arduino, ATMega328 port manipulation -ohje. Verkkosivu. 05.12.2023 https://friedchips.io/en/Arduino,+ATMega328+port+manipulation+pin+port+mapping.html

LIITTEET

Arduino koodi

```
#include <Wire.h>
#include <LCD_I2C.h>
#include <EEPROM.h>
#include <TimerOne.h>

const int yellowButton = 9;
const int redButton = 3;
const int whiteButton = 11;

// menu
int menu = 0;
const int maxMenu = 2;

// Pelien muuttujia
volatile bool enter = false;
volatile bool spedeLooper = true;
volatile bool spedeButtonPressed = true;
volatile unsigned long Timer1Int = 0;
unsigned long currentMS = 0;
unsigned long previousMS = millis();
volatile unsigned long startTime = 0;


// EEPROM
int s_address1 = 0;
int s_address2 = 1;
int s_address3 = 2;

// lcd
LCD_I2C lcd(0x27,20,4);

// Valosensori
uint16_t reading = 0;
int sensorPin = A7;


void setup() {

  DDRB &= ~2; // D9, yellow
  DDRB &= ~8; // D11, white
  DDRD &= ~8; // D3, red

  PORTB |= 2; // D9, input_pullup
  PORTB |= 8; // D11, input_pullup
  PORTD |= 8; // D3, input_pullup
```

```
  // Timer1
  // sammuta keskeytykset asennuksen ajaksi
  cli();
  // nollaa timer1
  TCCR1A = 0x00;
  TCCR1B = 0x00;

  TCNT1 = 0x0000;
  OCR1A = 16000;
  // aseta oikeat liput ja prescaler 1024
  TCCR1B |= (1 << WGM12);
  TCCR1B |= (1 << CS12) | (0 << CS11) | (1 << CS10) ;
  TIFR1 |= (1 << OCF1A);
  TIMSK1 |= (1 << OCIE1A);   // aktivoi keskeytyslippu
  sei();             // aktivoi keskeytykset

  // aseta lcd
  lcd.begin();
  lcd.clear();
  lcd.backlight();      // Make sure backlight is on

  // lisää interrupt punaiselle napille
  attachInterrupt(digitalPinToInterrupt(redButton), buttonISR, FALLING);

  // tulosta menu näytölle
  updateMenu(menu);

  // adc muuntimen setup
  ADMUX |= (1 << REFS0) | (1 << MUX2) | (1 << MUX1) | (1 << MUX0);
  ADCSRA |= (1 << ADEN) ;
  ADCSRB = 0x00;

  // EEPROMin tyhjäys tarvittaessa
  /*for (int i = 0 ; i < EEPROM.length() ; i++) {
    EEPROM.write(i, 0);
  }*/
}

void loop() {
  // lisää pieni debounce bufferi menuun
  currentMS = millis();
  if (currentMS - previousMS >= 150) {
    previousMS = currentMS;

    // jos nappikeskeytyksestä (punainen nappi) on saatu
    if (enter) {
      selectGame(menu);
    }

    // liikuttaa menua riippuen painaako keltaista tai valkoista
    if (!(PINB & 2)) {
```

```
      if (menu <= maxMenu && menu > 0) {
        menu--;
      }
      updateMenu(menu);
    }

    if (!(PINB & 8)) {
      if (menu < maxMenu && menu >= 0) {
        menu++;
      }
      updateMenu(menu);
    }
  }
}


// nappiin liitetyn keskeytyksen käyttämä aliohjelma
void buttonISR() {
  enter = true;

}


void selectGame(int gameNumber) {
  // Kutsu tiettyä peliä tietyn valikon numeron mukaan
  switch (gameNumber) {
    case 0:
      playSpede();
      enter = false;
      updateMenu(menu);
      break;
    case 1:
      playSakari();
      enter = false;
      updateMenu(menu);
      break;
    case 2:
      playInnerClockGame();
      enter = false;
      updateMenu(menu);
      break;
  }
}

void updateMenu(int position) {
  // tulosta valinnan mukaan tietty valinta näytölle
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Hikareiden konsoli ");

  switch (position) {
```

```
    case 0:
      lcd.setCursor(0, 1);
      lcd.print("   * Spede        *");
      lcd.setCursor(0, 2);
      lcd.print("     Sakari        ");
      lcd.setCursor(0, 3);
      lcd.print("     Minuuttipeli  ");
      break;
    case 1:
      lcd.setCursor(0, 1);
      lcd.print("     Spede         ");
      lcd.setCursor(0, 2);
      lcd.print("   * Sakari        *");
      lcd.setCursor(0, 3);
      lcd.print("     Minuuttipeli  ");
      break;
    case 2:
      lcd.setCursor(0, 1);
      lcd.print("     Spede         ");
      lcd.setCursor(0, 2);
      lcd.print("     Sakari        ");
      lcd.setCursor(0, 3);
      lcd.print("   * Minuuttipeli *");
      break;
  }
}

uint16_t ADC_read()
{
  //aloitetaan muunnos
  ADCSRA |= (1 << ADSC);
  // tehdään muunnos
  while (ADCSRA & B01000000)
  {
    ;
  }
  reading = ADC;
  // palautetaan muunnos
  return reading;
}

ISR(TIMER1_COMPA_vect)
{
  if (!spedeButtonPressed) {
    spedeLooper = !spedeLooper;
  }
  spedeButtonPressed = false;
}

// tarkista onko top3 ja muokkaa tarvittaessa EEPROM arvoja
bool chech_high_score( int score) {
```

```
  bool was_top = false;
  int top1 = 0;
  int top2 = 0;
  int top3 = 0;
  EEPROM.get(s_address1, top1);
  EEPROM.get(s_address2, top2);
  EEPROM.get(s_address3, top3);

  // jos top1 sija ei ole tyhjä tai top1 on pienempi kuin uusi tulos
  if (top1 <= score) {
    // jos top1 pienempi kuin score siirrä score ekaan sijaan
    // ja siirrä loput 1 sija alaspäin
    EEPROM.put(s_address1, score);
    EEPROM.put(s_address2, top1);
    EEPROM.put(s_address3, top2);
    was_top = true;
  // jos top2 on pienempi kuin uusi tulos
  // siirrä top2 -> top3
    } else if (top2 <= score) {
    EEPROM.put(s_address2, score);
    EEPROM.put(s_address3, top2);
    was_top = true;
    // jos to3 sija
  } else if (top3 < score) {
    EEPROM.put(s_address3, score);
    was_top = true;
  }
  // palauta oliko top3 vai eikö
  return was_top;
}

void playSakari() {
  // aseta tarvittavat muuttujat
  currentMS = 0;
  previousMS = millis();
  int start_count_down = 10;
  bool win = false;
  bool countDownLooper = true;

  // tulosta näytölle
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("  Pue villapaita?");

  // tutki valoanturin arvoa ADC_readilla ja
  // 10 sec ajan. Jos villapaitaa ei ole puettu
  // 10 sekunnissa, häviää ja jos reading on alle 20
  // eli paita on puettu, se on voitto
  currentMS = millis();
  while (countDownLooper) {
    reading = ADC_read();
```

```
    if (reading < 20) {
      countDownLooper = false;
      win = true;
    }
    currentMS = millis();
    if (currentMS - previousMS >= 1000) {
      previousMS = currentMS;
      start_count_down--;
      lcd.setCursor(9, 3);
      lcd.print(start_count_down);
      lcd.print(" ");
      if (start_count_down == 0) {
        countDownLooper = false;
      }
    }
  }
  countDownLooper = true;
  // tulostaa oikean tuloksen jos puettu/ei puettu
  if (win) {
    lcd.clear();
    lcd.setCursor(5, 1);
    lcd.print("jee puettu");
  } else {
    lcd.clear();
    lcd.setCursor(7, 1);
    lcd.print("HYRRRRR");
  }
  delay(3000);
}

void playInnerClockGame() {
  // tulostaa aloitusnäytön käytön
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("INNER CLOCK GAME");
  lcd.setCursor(0, 1);
  lcd.print("PRESS WHITE TO START");

  // luo seuraavaa while-loopia varten kontrollointi-muuttujat
  bool looper = true;
  bool clickedButton = false;
  int counter = 1;

  while (looper) {
    // muokkaa arvoja ja tulostaa näytölle tulosteet vain kerran
    // nappia painettua ja counterin ollessa 1
    if (!(PINB & 8) && counter == 1) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("COUNT MINUTE THEN");
      lcd.setCursor(0, 1);
```

```
      lcd.print("RELEASE WHITE        ");
      lcd.setCursor(0, 2);
      lcd.print("TO STOP");
      counter++;
      startTime = millis();
      clickedButton = true;
    // jos on jo painettu ja nappi nousee looppi katkeaa
    } else if ((PINB & 8) && clickedButton) {
      looper = false;
    }

  }
  // tulostaa scoren näytölle
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("YOUR SCORE:");
  lcd.setCursor(0, 1);
  displayScore();
  delay(3000);
}

void displayScore() {

  unsigned long currentTime = millis();
  unsigned long elapsedTime = (currentTime - startTime) / 10; // divide
by 10 to get two decimal places

  int seconds = elapsedTime / 100;
  int milliseconds = elapsedTime % 100;

  lcd.setCursor(0, 3);
  lcd.print("Time: " + String(seconds) + "." + String(milliseconds) +
"s");

  // Calculate the score based on proximity to 60 seconds
  int score = 100 - abs(60 - seconds);
  score = max(0, score); // Ensure the score is not negative

  lcd.setCursor(0, 2);
  lcd.print("Score: " + String(score) + "P");
}

void playSpede() {
  // luo ja alustaa peliin tarvittavat muuttujat
  int score = 0;
  bool countDownLooper = true;
  int start_count_down = 3;
  int random_button = 0;
  currentMS = 0;
  previousMS = millis();
```

```
  // tulosta pelin tiedot ja 3 sekunnin countdown
  lcd.clear();
  lcd.setCursor(7, 0);
  lcd.print("SPEDE");
  lcd.setCursor(9, 2);
  lcd.print(start_count_down);

  while (countDownLooper) {
    currentMS = millis();
    if (currentMS - previousMS >= 1000) {
      previousMS = currentMS;
      start_count_down--;
      lcd.setCursor(9, 2);
      lcd.print(start_count_down);
      if (start_count_down == 0) {
        countDownLooper = false;
      }
    }
  }

  // alusta timer1 aika
  Timer1Int = 16000;
  // tarvittavat muuttujat loopin hallintaan
  spedeLooper = true;
  spedeButtonPressed = true;

  random_button = random(1, 4);

  bool yellowButtonUp = true;
  bool whiteButtonUp = true;
  bool redButtonUp = true;

  while (spedeLooper) {
    int bounce = 60;
    // aja Timer1Int arvo sisälle timer1 niin, että saa timerin pienene-
määän
    OCR1A = Timer1Int;

    // tarkista onko napit nostettu
    if (PINB & 8){
      whiteButtonUp = true;
    }
    if (PIND & 8){
      redButtonUp = true;
    }
    if (PINB & 2){
      yellowButtonUp = true;
    }
    // jos kaikki napit on nostettu, tarkista random numero ja päivitä
sen
    // mukaan pisteitä ja näytön tulostetta
```

```
    // HUOM. taustalla pauhaa timer1 joka hallitsee spedeButtonPressed ja
spedeLooper arvoa
    if (yellowButtonUp && whiteButtonUp && redButtonUp) {
      if (random_button == 1) {
        lcd.setCursor(0, 2);
        lcd.print("  X  |       |      ");
        lcd.setCursor(6, 4);
        lcd.print("Score: ");
        lcd.print(score);

        currentMS = millis();
        if (currentMS - previousMS >= bounce) {
          previousMS = currentMS;
          if (!(PINB & 2)) {
            spedeButtonPressed = true;
            random_button = random(1, 4);
            score++;
            Timer1Int -= 500;
            yellowButtonUp = false;
          } else if ((!(PINB & 8)) || (!(PIND & 8))){
            spedeLooper = false;
          }
        }

      } else if (random_button == 2) {
        lcd.setCursor(0, 2);
        lcd.print("     |   X   |      ");
        lcd.setCursor(6, 4);
        lcd.print("Score: ");
        lcd.print(score);
        currentMS = millis();
        if (currentMS - previousMS >= bounce) {
          previousMS = currentMS;
          if (!(PIND & 8)) {
            spedeButtonPressed = true;
            random_button = random(1, 4);
            score++;
            Timer1Int -= 500;
            redButtonUp = false;
          } else if ((!(PINB & 2)) || (!(PINB & 8))){
            spedeLooper = false;
          }
        }

      } else if (random_button == 3) {
        lcd.setCursor(0, 2);
        lcd.print("     |       |  X   ");
        lcd.setCursor(6, 4);
        lcd.print("Score: ");
        lcd.print(score);
        currentMS = millis();
```

```
        if (currentMS - previousMS >= bounce) {
          previousMS = currentMS;
          if (!(PINB & 8)) {
            spedeButtonPressed = true;
            random_button = random(1, 4);
            score++;
            Timer1Int -= 500;
            whiteButtonUp = false;
          } else if ((!(PINB & 2)) || (!(PIND & 8))){
            spedeLooper = false;
          }
        }
      }
    }
  }
  // tarkista chech_high_score-aliohjelmaa käyttäen onko
  // score top3 ja ota EEPPROMista tulostusta varten tiedot
  bool high_score = chech_high_score(score);
  int top1 = EEPROM.read(s_address1);
  int top2 = EEPROM.read(s_address2);
  int top3 = EEPROM.read(s_address3);

  lcd.clear();
  lcd.setCursor(0, 0);
  // tulosta sen mukaan oliko top3 3 sekunnin ajan
  if (high_score) {
    lcd.print("VICTORY! SCORE: ");
  } else {
    lcd.print("NOT TOP3. SCORE: ");
  }
  lcd.print(score);
  lcd.setCursor(7, 1);
  lcd.print("TOP1: ");
  lcd.print(top1);
  lcd.setCursor(7, 2);
  lcd.print("TOP2: ");
  lcd.print(top2);
  lcd.setCursor(7, 3);
  lcd.print("TOP3: ");
  lcd.print(top3);

  delay(3000);
}
```